


```
SSSSSSSS RRRRRRRR CCCCCCCC VV VV 000000 LL
SSSSSSSS RRRRRRRR CCCCCCCC VV VV 000000 LL
SS      RR      RR CC CC VV VV 00 00 LL
SS      RR      RR CC CC VV VV 00 00 LL
SS      RR      RR CC CC VV VV 00 00 LL
SSSSSS RRRRRRRR CCCCCCCC VV VV 00 00 LL
SSSSSS RRRRRRRR CCCCCCCC VV VV 00 00 LL
SS      RR      RR CC CC VV VV 00 00 LL
SS      RR      RR CC CC VV VV 00 00 LL
SS      RR      RR CC CC VV VV 00 00 LL
SSSSSSSS RRR      RR      CCCCCCCC VV VV 000000 LLLLLLLLLL
SSSSSSSS RRR      RR      CCCCCCCC VV VV 000000 LLLLLLLLLL
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      IIIIII SSSSSSSS
LL      IIIIII SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```



```
1 0001 0 MODULE SRCVOL (  
2 0002 0     LANGUAGE (BLISS32),  
3 0003 0     IDENT = 'V04-000',  
4 0004 0     ) =  
5 0005 1 BEGIN  
6 0006 1  
7 0007 1  
8 0008 1 *****  
9 0009 1 *  
10 0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
11 0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
12 0012 1 *  ALL RIGHTS RESERVED.  
13 0013 1 *  
14 0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
15 0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
16 0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
17 0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
18 0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
19 0019 1 *  TRANSFERRED.  
20 0020 1 *  
21 0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
22 0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
23 0023 1 *  CORPORATION.  
24 0024 1 *  
25 0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
26 0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
27 0027 1 *  
28 0028 1 *  
29 0029 1 *****  
30 0030 1  
31 0031 1 ++  
32 0032 1  
33 0033 1 FACILITY: MOUNT Utility Structure Level 1  
34 0034 1  
35 0035 1 ABSTRACT:  
36 0036 1  
37 0037 1     This routine searches the device database for a particular volume.  
38 0038 1  
39 0039 1 ENVIRONMENT:  
40 0040 1  
41 0041 1     STARLET operating system, including privileged system services  
42 0042 1     and internal exec routines.  
43 0043 1  
44 0044 1 --  
45 0045 1  
46 0046 1  
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 13-Oct-1977 20:09  
48 0048 1  
49 0049 1 MODIFIED BY:  
50 0050 1  
51 0051 1     V03-008 HH0050     Hai Huang     17-Aug-1984  
52 0052 1     Return success status if MOUNT/CLUSTER on a volume  
53 0053 1     that is already mounted locally.  
54 0054 1  
55 0055 1     V03-007 HH0041     Hai Huang     24-Jul-1984  
56 0056 1     Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.  
57 0057 1
```

SRCVOL
V04-000

M 15

16-Sep-1984 01:33:17
14-Sep-1984 12:45:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]SRCVOL.B32;1

Page 2
(1)

```

58 0058 1  V03-006 HH0036 Hai Huang 11-Jul-1984
59 0059 1  Allow MOUNT/CLUSTER on a volume that is already mounted
60 0060 1  on the local node.
61 0061 1
62 0062 1  V03-005 HH0024 Hai Huang 18-Jun-1984
63 0063 1  Call IOC$CVT_DEVNAM to format the device name on a
64 0064 1  successful shared mount.
65 0065 1
66 0066 1  V03-004 HH0002 Hai Huang 01-Feb-1984
67 0067 1  Add job-wide mount support, i.e. deallocate mount list
68 0068 1  entries to paged-pool.
69 0069 1
70 0070 1  V03-003 CDS0001 Christian D. Saether 17-Jun-1983
71 0071 1  Fix SEARCH_VOL so it knows how to run through
72 0072 1  all system blocks, and therefore find all ddb's.
73 0073 1
74 0074 1  V03-002 DMW4041 DMWalp 7-Jun-1983
75 0075 1  Remove (S)LOG_ENTRY
76 0076 1
77 0077 1  V03-001 STJ50311 Steven T. Jeffreys, 11-Feb-1983
78 0078 1  Make all uses of PHYS_NAME indexed by DEVICE_INDEX.
79 0079 1
80 0080 1  V02-004 STJ0195 Steven T. Jeffreys, 02-Feb-1982
81 0081 1  Removed SYSEXV declaration.
82 0082 1
83 0083 1  V02-003 STJ0165 Steven T. Jeffreys, 04-Jan-1982
84 0084 1  Remove obsolete calls to $SETEXV.
85 0085 1
86 0086 1  V02-002 ACG0169 Andrew C. Goldstein, 18-Apr-1980 13:58
87 0087 1  Bug check on internal errors
88 0088 1
89 0089 1  V02-001 ACG0167 Andrew C. Goldstein, 18-Apr-1980 13:39
90 0090 1  Previous revision history moved to MOUNT.REV
91 0091 1  **
92 0092 1
93 0093 1
94 0094 1  LIBRARY 'SYSS$LIBRARY:LIB.L32';
95 0095 1  REQUIRE 'SRC$:MOUDEF.B32';
96 0627 1
97 0628 1
98 0629 1  FORWARD ROUTINE
99 0630 1  SEARCH_VOL, ! search for mounted volume
100 0631 1  SEARCH_HANDLER : NOVALUE; ! condition handler for above
```



```
0632 1 GLOBAL ROUTINE SEARCH_VOL (NAME) =
0633 1
0634 1 ++
0635 1
0636 1 FUNCTIONAL DESCRIPTION:
0637 1
0638 1     This routine searches the device database for a particular volume.
0639 1     Only file structured devices mounted /SYSTEM, /GROUP, or /SHARE
0640 1     are considered.
0641 1
0642 1
0643 1 CALLING SEQUENCE:
0644 1     SEARCH_VOL (ARG1)
0645 1
0646 1 INPUT PARAMETERS:
0647 1     ARG1: string descriptor of volume label
0648 1
0649 1 IMPLICIT INPUTS:
0650 1     MOUNT_OPTIONS: command option bits
0651 1
0652 1 OUTPUT PARAMETERS:
0653 1     NONE
0654 1
0655 1 IMPLICIT OUTPUTS:
0656 1     NONE
0657 1
0658 1 ROUTINE VALUE:
0659 1     UCB address of found volume or zero
0660 1
0661 1 SIDE EFFECTS:
0662 1     mount count of found volume is incremented if /SHARE mode mount
0663 1
0664 1 --
0665 1
0666 2 BEGIN
0667 2
0668 2 MAP
0669 2     NAME          : REF VECTOR;    ! volume name string descriptor
0670 2
0671 2 LABEL
0672 2     SEARCH_LOOP;    ! outer device search loop
0673 2
0674 2 LOCAL
0675 2     STATUS,        ! random status value
0676 2     SB              : REF BBLOCK,    ! pointer to current SB
0677 2     DDB             : REF BBLOCK,    ! pointer to current DDB
0678 2     UCB             : REF BBLOCK,    ! pointer to current UCB
0679 2     VCB             : REF BBLOCK;    ! pointer to current VCB
0680 2
0681 2 EXTERNAL
0682 2     MOUNT_OPTIONS  : BITVECTOR,      ! MOUNT command options
0683 2     DEVICE_INDEX   : LONG,           ! index into PHYS_NAME vector
0684 2     PHYS_NAME      : VECTOR,         ! physical device name descriptor
0685 2     NAME_BUFFER    : VECTOR [BYTE], ! physical device name buffer
0686 2     MTL_ENTRY      : REF BBLOCK,     ! address of mount list entry
0687 2     SMT_ENTRY      : REF BBLOCK,     ! address of volume set mount list entry
0688 2     SCSSGQ_CONFIG  : ADDRESSING_MODE (ABSOLUTE),
```



```
159 0689 2      MOUNT_ITMLST : LONG;      ! I/O database listhead
160 0690 2      ! address of the mount item list
161 0691 2
162 0692 2 LINKAGE
163 0693 2      IOC_CVT_DEVNAM = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 4,
164 0694 2      REGISTER = 5, REGISTER = 1) :
165 0695 2      PRESERVE (2,3,4,5,6,7);
166 0696 2
167 0697 2 EXTERNAL ROUTINE
168 0698 2      LOCK_IODB,      ! lock I/O database mutex
169 0699 2      UNLOCK_IODB,    ! unlock I/O database mutex
170 0700 2      DEALLOCATE_MEM, ! deallocate dynamic memory
171 0701 2      ALLOC_LOGNAME,   ! create logical name and MTL entry
172 0702 2      ENTER_LOGNAME,  ! hook up logical name and MTL entry
173 0703 2      IOC$CVT_DEVNAM : IOC_CVT_DEVNAM ADDRESSING_MODE (GENERAL),
174 0704 2      ! get fully expanded device name
175 0705 2      MOUNT_CLUSTER;    ! mount volume cluster-wide
176 0706 2
177 0707 2
178 0708 2 ! Enable our condition handler.
179 0709 2 ! Needless to say, the search must be done with the I/O database locked to
180 0710 2 ! prevent list perturbations. We run down the DDB list, following the UCB
181 0711 2 ! list off each one, looking for file structured devices that are mounted
182 0712 2 ! but not allocated.
183 0713 2 !
184 0714 2
185 0715 2 ENABLE SEARCH_HANDLER;
186 0716 2
187 0717 2 ! For a shared mount, preallocate logical name and mounted volume list entry
188 0718 2 ! for the worst case (RVN 1), since we cannot tolerate failures after bumping
189 0719 2 ! the mount count in a found VCB. If not needed, we discard them later. System
190 0720 2 ! and Group mounts don't need this, since we will fail here one way or the other.
191 0721 2 !
192 0722 2
193 0723 2 IF .MOUNT_OPTIONS[OPT_SHARE]
194 0724 2 THEN
195 0725 2 BEGIN
196 0726 2     ALLOC_LOGNAME (0);
197 0727 2     SMTL_ENTRY = .MTL_ENTRY;
198 0728 2     MTL_ENTRY = 0;
199 0729 2     ALLOC_LOGNAME (1);
200 0730 2 END;
201 0731 2
202 0732 2 LOCK_IODB ();
203 0733 2
204 0734 2 SB = .SCS$GQ_CONFIG;
205 0735 2
206 0736 2 SEARCH_LOOP:
207 0737 2 BEGIN
208 0738 2 UNTIL .SB EQL SCS$GQ_CONFIG
209 0739 2 DO
210 0740 2 BEGIN
211 0741 2
212 0742 2 DDB = .SB [SB$L_DDB];
213 0743 2
214 0744 2 WHILE .DDB NEQ 0
215 0745 2 DO
```



```
216 0746 5 BEGIN
217 0747 5 IF .DDB[DDB$B_TYPE] NEQ DYN$C_DDB
218 0748 5 THEN BUG_CHECK (NOTDDBDDB, FATAL, 'Corrupted DDB List');
219 0749 5 UCB = .DDB[DDB$L_UCB];
220 0750 5
221 0751 5 UNTIL .UCB EQL 0 DO
222 0752 6 BEGIN
223 0753 6 IF .UCB[UCB$B_TYPE] NEQ DYN$C_UCB
224 0754 6 THEN BUG_CHECK (NOTUCBUCEB, FATAL, 'Corrupted UCB List');
225 0755 6
226 0756 6 IF .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_FOD]
227 0757 6 AND .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_MNT]
228 0758 6 AND NOT .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_ALL]
229 0759 6 AND NOT .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_DMT]
230 0760 6 THEN
231 0761 7 BEGIN
232 0762 7 VCB = .UCB[UCB$L_VCB];
233 0763 7 IF .VCB[VCB$B_TYPE] NEQ DYN$C_VCB
234 0764 7 THEN BUG_CHECK (NOTVCBUCEB, FATAL, 'Not VCB pointer in UCB');
235 0765 7 IF CH$EQ (.NAME[0], .NAME[1],
236 0766 7 VCB$S_VOLNAME, VCB[VCB$T_VOLNAME], ' ')
237 0767 7 THEN LEAVE SEARCH_LOOP;
238 0768 6 END;
239 0769 6 UCB = .UCB[UCB$L_LINK];
240 0770 5 END; ! end of UCB scan loop
241 0771 5 DDB = .DDB[DDB$L_LINK];
242 0772 4 END; ! end of DDB scan loop
243 0773 4
244 0774 4 SB = .SB [SB$L_FLINK];
245 0775 3 END;
246 0776 2 END; ! end of block SEARCH_LOOP
247 0777 2
248 0778 2 ! If we find a suitable volume that matches the name, the search is over.
249 0779 2 ! If the mount is /SHARE, we increment the volume mount count here and now
250 0780 2 ! to avoid race conditions. Finding a volume on a /SYSTEM or /GROUP mount
251 0781 2 ! is an error.
252 0782 2
253 0783 2
254 0784 2 STATUS = 0;
255 0785 2 IF .UCB NEQ 0
256 0786 2 THEN
257 0787 3 BEGIN
258 0788 3 IF .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_FOR] NEQ .MOUNT_OPTIONS[OPT_FOREIGN]
259 0789 3 THEN STATUS = MOUN$_INCOMPAT
260 0790 3
261 0791 3 ELSE
262 0792 4 BEGIN
263 0793 4 IF NOT .MOUNT_OPTIONS[OPT_SHARE]
264 0794 4 THEN
265 0795 4 STATUS = MOUN$_VOLALRMNT
266 0796 4 ELSE
267 0797 4 BEGIN
268 0798 5 STATUS = 1;
269 0799 5 VCB[VCB$W_MCOUNT] = .VCB[VCB$W_MCOUNT] + 1;
270 0800 5 END;
271 0801 4 !
272 0802 4
```



```
273 0803 4      ! For a successful shared mount or volume already mounted, format the
274 0804 4      ! expanded device name and set up the device name descriptor.
275 0805 4
276 0806 4      !OC$CVT_DEVNAM (NAMEBUF_LEN,          ! Output buffer length
277 0807 4      NAME_BUFFER +
278 0808 4      (.DEVICE_INDEX*NAMEBUF_LEN),      ! Output buffer address
279 0809 4      -1,                                ! Format device name in display form
280 0810 4      .UCB;                             ! Address of the target UCB
281 0811 4      PHYS_NAME [.DEVICE_INDEX*2]);      ! Returned length
282 0812 4      PHYS_NAME[(.DEVICE_INDEX*2)+1] = NAME_BUFFER +
283 0813 4      (.DEVICE_INDEX*NAMEBUF_LEN);      ! Set up device name descriptor
284 0814 3      END;
285 0815 2      END;
286 0816 2
287 0817 2      UNLOCK_IODB ();
288 0818 2
289 0819 2      !
290 0820 2      ! If the /CLUSTER option is specified, send the request cluster-wide even
291 0821 2      ! if the volume is already mounted on the local node. Return with success
292 0822 2      ! in this case.
293 0823 2
294 0824 3      IF ( .STATUS EQL MOUN$ VOLALRMNT)
295 0825 3      AND ( .MOUNT_OPTIONS [OPT_CLUSTER])
296 0826 2      THEN
297 0827 3      BEGIN
298 0828 3      MOUNT_CLUSTER (.MOUNT_ITMLST);
299 0829 3      RETURN 1;
300 0830 2      END;
301 0831 2
302 0832 2      IF NOT .STATUS THEN ERR_EXIT (.STATUS);
303 0833 2
304 0834 2      !
305 0835 2      ! We now have a successful shared mount. Fill in the logical name and MTL
306 0836 2      ! entries.
307 0837 2
308 0838 2      ! If what we found is RVN 1 of a volume set, keep the entries as is. Otherwise
309 0839 2      ! release one of each.
310 0840 2
311 0841 2      !
312 0842 2      IF .BBLOCK [UCB[UCB$L DEVCHAR], DEV$V_SQD]
313 0843 2      OR .VCB[VCB$W_RVN] NEQ 1
314 0844 2      THEN
315 0845 3      BEGIN
316 0846 3      DEALLOCATE_MEM (.MTL_ENTRY, 1);
317 0847 3      MTL_ENTRY = .SMTL_ENTRY;
318 0848 3      SMTL_ENTRY = 0;
319 0849 3      END;
320 0850 2
321 0851 2      ENTER_LOGNAME (.UCB, .VCB);
322 0852 2
323 0853 2      RETURN 1;
324 0854 2
325 0855 1      END;

! end of routine SEARCH_VOL
```

```
.TITLE SRCVOL
.IDENT \V04-000\
```


50	01	AA	55	30	A5	D0	00091	8\$:	MOVL	48(UCB), UCB	:	0769		
50	3B	A5	54		BE	11	00095		BRB	5\$	:	0751		
			57		64	D0	00097	9\$:	MOVL	(DDB), DDB	:	0771		
					A9	11	0009A		BRB	3\$	:	0744		
					67	D0	0009C	10\$:	MOVL	(SB), SB	:	0774		
					9B	11	0009F		BRB	2\$	:	0738		
					52	D4	000A1	11\$:	CLRL	STATUS	:	0784		
					55	D5	000A3		TSTL	UCB	:	0785		
					5D	13	000A5		BEQL	15\$	:			
			01		03	EF	000A7		EXTZV	#3, #1, MOUNT_OPTIONS+1, R0	:	0788		
			01		00	ED	000AD		CMPZV	#0, #1, 59(UCB), R0	:			
					09	13	000B3		BEQL	12\$	:			
			52	007280AC	8F	D0	000B5		MOVL	#7504044, STATUS	:	0789		
					46	11	000BC		BRB	15\$	:			
		09	6A		06	E0	000BE	12\$:	BBS	#6, MOUNT_OPTIONS, 13\$	:	0794		
			52	007280B4	8F	D0	000C2		MOVL	#7504052, STATUS	:	0796		
					06	11	000C9		BRB	14\$	:			
			52		01	D0	000CB	13\$:	MOVL	#1, STATUS	:	0799		
				4C	A6	B6	000CE		INCW	76(VCB)	:	0800		
			50	0000G	CF	05	78	000D1	14\$:	ASHL	#5, DEVICE_INDEX, R0	:	0808	
			51		0000GCF	40	9E	000D7		MOVAB	NAME_BUFFER[R0], R1	:	0807	
			54		01	CE	000DD		MNEGL	#1, R4	:	0811		
			50		20	D0	000E0		MOVL	#32, R0	:			
				00000000G	00	16	000E3		JSB	IOC\$CVT_DEVNAM	:			
			50	0000G	CF	01	78	000E9		ASHL	#1, DEVICE_INDEX, R0	:		
				0000GCF	40	51	D0	000EF		MOVL	R1, PHYS_NAME[R0]	:		
			51	0000G	CF	05	78	000F5		ASHL	#5, DEVICE_INDEX, R1	:	0813	
				0000GCF	40	41	9E	000FB		MOVAB	NAME_BUFFER[R1], PHYS_NAME+4[R0]	:	0812	
				0000G	CF	00	FB	00104	15\$:	CALLS	#0, UNLOCK_IOCB	:	0817	
			007280B4	8F	52	D1	00109		CMPPL	STATUS, #7504052	:	0824		
					10	12	00110		BNEQ	16\$	:			
			0B	07	AA	06	E1	00112		BBC	#6, MOUNT_OPTIONS+7, 16\$	:	0825	
					0000G	CF	DD	00117		PUSHL	MOUNT_ITMEST	:	0828	
				0000G	CF	01	FB	0011B		CALLS	#1, MOUNT_CLUSTER	:		
					2D	11	00120		BRB	20\$	:	0829		
			09		52	E8	00122	16\$:	BLBS	STATUS, 17\$	:	0832		
					52	DD	00125		PUSHL	STATUS	:			
				00000000G	00	01	FB	00127		CALLS	#1, LIB\$STOP	:		
			06	38	A5	05	E0	0012E	17\$:	BBS	#5, 56(UCB), 18\$	:	0842	
					01	0E	A6	B1	00133		CMPW	14(VCB), #1	:	0843
						0E	13	00137		BEQL	19\$	:		
					01	DD	00139	18\$:	PUSHL	#1	:	0846		
					68	DD	0013B		PUSHL	MTL_ENTRY	:			
				0000G	CF	02	FB	0013D		CALLS	#2, DEALLOCATE_MEM	:		
				68	6B	D0	00142		MOVL	SMTL_ENTRY, MTC_ENTRY	:	0847		
					6B	D4	00145		CLRL	SMTL_ENTRY	:	0848		
				7E	55	7D	00147	19\$:	MOVQ	UCB, -(SP)	:	0851		
				0000G	CF	02	FB	0014A		CALLS	#2, ENTER_LOGNAME	:		
				50	01	D0	0014F	20\$:	MOVL	#1, R0	:	0853		
						04	00152		RET		:	0855		
						0000	00153	21\$:	.WORD	Save nothing	:	0666		
					7E	D4	00155		CLRL	-(SP)	:			
					5E	DD	00157		PUSHL	SP	:			
				7E	AC	7D	00159		MOVQ	4(AP), -(SP)	:			
			0000V	CF	03	FB	0015D		CALLS	#3, SEARCH_HANDLER	:			
					04	00162			RET		:			

SRCVOL
V04-000

G 16
16-Sep-1984 01:33:17
14-Sep-1984 12:45:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]SRCVOL.B32;1 Page 9 (2)

; Routine Size: 355 bytes, Routine Base: \$CODE\$ + 0000


```
0856 1 ROUTINE SEARCH_HANDLER (SIGNAL, MECHANISM) : NOVALUE =
0857 1
0858 1 ++
0859 1
0860 1 FUNCTIONAL DESCRIPTION:
0861 1
0862 1 This routine is the condition handler for the volume search
0863 1 routine. It undoes any damage done so far and returns the error
0864 1 status to the user mode caller.
0865 1
0866 1
0867 1 CALLING SEQUENCE:
0868 1 KERNEL_HANDLER (ARG1, ARG2)
0869 1
0870 1 INPUT PARAMETERS:
0871 1 ARG1: address of signal vector
0872 1 ARG2: address of mechanism vector
0873 1
0874 1 IMPLICIT INPUTS:
0875 1 global pointers to blocks allocated
0876 1
0877 1 OUTPUT PARAMETERS:
0878 1 NONE
0879 1
0880 1 IMPLICIT OUTPUTS:
0881 1 NONE
0882 1
0883 1 ROUTINE VALUE:
0884 1 NONE
0885 1
0886 1 SIDE EFFECTS:
0887 1 stack unwound, allocations undone
0888 1
0889 1 --
0890 1
0891 2 BEGIN
0892 2
0893 2 MAP
0894 2 SIGNAL : REF BBLOCK, ! signal vector
0895 2 MECHANISM : REF BBLOCK; ! mechanism vector
0896 2
0897 2 EXTERNAL
0898 2 MOUNT_OPTIONS : BITVECTOR, ! command parser options
0899 2 MTL_ENTRY : REF BBLOCK, ! address of mount list entry
0900 2 SMT_ENTRY : REF BBLOCK; ! address of volume set mount list entry
0901 2
0902 2 EXTERNAL ROUTINE
0903 2 UNLOCK_IODB, ! unlock I/O database mutex
0904 2 DEALLOCATE_MEM; ! deallocate system dynamic memory
0905 2
0906 2
0907 2 IF .SIGNAL[CHFSL_SIG_NAME] NEQ SSS_UNWIND
0908 2 THEN
0909 2 BEGIN
0910 2
0911 2 IF .SIGNAL[CHFSL_SIG_ARGS] NEQ 3
0912 2 THEN BUG_CHECK (ONXSIGNAL, FATAL, 'Unexpected signal in MOUNT');
```



```
0913 384  
0914 385  
0915 386  
0916 387  
0917 388  
0918 389  
0919 390  
0920 391  
0921 392  
0922 393  
0923 394  
0924 395  
0925 396  
0926 397  
0927 398  
0928 399  
0929 400  
0930 401  
0931 402  
0932 403  
0933 404
```

```
! Deallocate whatever control blocks exist to wherever they came from.  
IF .MTL_ENTRY NEQ 0  
THEN DEALLOCATE_MEM (.MTL_ENTRY, 1);  
IF .SMTL_ENTRY NEQ 0  
THEN DEALLOCATE_MEM (.SMTL_ENTRY, 1);  
MTL_ENTRY = SMTL_ENTRY = 0;  
! Return the condition code in R0.  
MECHANISM[CHF$MCH_SAVRO] = .SIGNAL[CHF$SIG_NAME];  
$UNWIND ();  
END;  
! end of routine KERNEL_HANDLER
```

```
.EXTRN BUG$_UNXSIGNAL, SYSSUNWIND  
0000 00000 SEARCH_HANDLER:  
00000920 50 04 AC D0 00002 .WORD Save nothing  
8F 04 A0 D1 00006 MOVL SIGNAL, R0  
43 13 0000E CMPL 4(R0), #2336  
03 60 D1 00010 BEQL 4$  
04 13 00013 CMPL (R0), #3  
FEFF 00015 BEQL 1$  
0000* 00017 BUGW  
50 0000G CF D0 00019 1$: .WORD <BUG$ UNXSIGNAL!4>  
09 13 0001E MOVL MTL_ENTRY, R0  
01 DD 00020 BEQL 2$  
50 DD 00022 PUSHL #1  
0000G CF 02 FB 00024 PUSHL R0  
50 0000G CF D0 00029 2$: CALLS #2, DEALLOCATE_MEM  
09 13 0002E MOVL SMTL_ENTRY, R0  
01 DD 00030 BEQL 3$  
50 DD 00032 PUSHL #1  
0000G CF 02 FB 00034 PUSHL R0  
0000G CF 04 00039 3$: CALLS #2, DEALLOCATE_MEM  
0000G CF D4 0003D CLRL SMTL_ENTRY  
50 04 AC 7D 00041 CLRL MTL_ENTRY  
0C A1 04 A0 D0 00045 MOVQ SIGNAL, R0  
7E 7C 0004A MOVL 4(R0), 12(R1)  
00000000G 00 02 FB 0004C CLRQ -(SP)  
04 00053 4$: CALLS #2, SYSSUNWIND  
RET
```

0856
0907
0911
0912
0918
0919
0921
0922
0924
0929
0930
0933

; Routine Size: 84 bytes, Routine Base: \$CODE\$ + 0163

; 405 0934 1

SRCVOL
V04-000

J 16
16-Sep-1984 01:33:17
14-Sep-1984 12:45:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]SRCVOL.B32;1 Page 12 (3)

: 406
: 407
: 0935 1 END
: 0936 0 ELUDOM

.EXTRN LIB\$STOP

PSECT SUMMARY

:
: Name Bytes Attributes
: \$CODE\$ 439 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

:
: File Total Symbols Loaded Percent Pages Mapped Processing Time
: _\$255\$DUA28:[SYSLIB]LIB.L32;1 18619 39 0 1000 00:01.9

COMMAND QUALIFIERS

:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SRCVOL/OBJ=OBJ\$:SRCVOL MSRC\$:SRCVOL/UPDATE=(ENH\$:SRCVOL)

: Size: 439 code + 0 data bytes
: Run Time: 00:16.4
: Elapsed Time: 00:43.9
: Lines/CPU Min: 3422
: Lexemes/CPU-Min: 29093
: Memory Used: 157 pages
: Compilation Complete

0246 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

